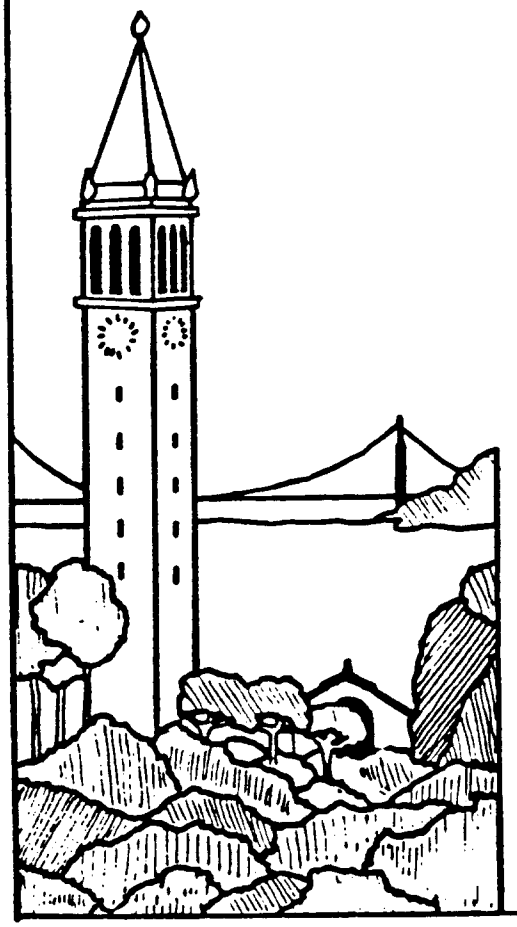
The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD

Riccardo Gusella and Stefano Zatti



Report No. UCB/CSD 87/337

January 1987
PROGRES Report No. 86.7

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JAN 1987		**00-00-1987 to 00-00-1987**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper discusses the upper and lower bounds on the accuracy of the time synchronization achieved by the algorithms implemented in TEMPO, a distributed clock synchronizer running on Berkeley UNIX 4.3BSD systems. We show that the accuracy is a function of the network transmission latency, and depends linearly upon the drift rate of the clocks and the interval between synchronizations. Comparison with other clock synchronization algorithms reveals that TEMPO may achieve better synchronization accuracy at a lower cost.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**16**	

The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD

Riccardo Gusella and Stefano Zatti[†]

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

ABSTRACT

This paper discusses the upper and lower bounds on the accuracy of the time synchronization achieved by the algorithms implemented in TEMPO, a distributed clock synchronizer running on Berkeley UNIX 4.3BSD systems. We show that the accuracy is a function of the the network transmission latency, and depends linearly upon the drift rate of the clocks and the interval between synchronizations. Comparison with other clock synchronization algorithms reveals that TEMPO may achieve better synchronization accuracy at a lower cost.

Introduction

This paper discusses the upper and lower bounds on the accuracy of the time synchronization achieved by the algorithms implemented in TEMPO, a distributed clock synchronizer running on Berkeley UNIX 4.3BSD systems.

TEMPO, which works in a local area network, consists of a collection of *time daemons* (one per machine) and is based on a master-slave structure[2,3].

This work was sponsored by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871 monitored by the Naval Electronics Systems Command under contract No. N00039-84-C-0089, and by the CSELT Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Research Projects Agency, of the US Government, or of CSELT.

† Author's current address: IBM Zurich Research Laboratory, Saumerstrasse 4, CH-8803 Rueschlikon, Switzerland.

Figures 1a and 1b sketch the way TEMPO works. A *master time daemon* measures the time difference between the clock of the machine on which it is running and those of all other machines. The master computes the *network time* as the average of the times provided by nonfaulty clocks‡. It then sends to each *slave time daemon* the correction that should be performed on the clock of its machine. Since the correction can be negative, in order to preserve the monotonicity of the clocks' time functions, TEMPO implements it by slowing down (or speeding up) the clock rates[1]. This process is repeated periodically. Because the correction is expressed as a time difference rather than an absolute time, transmission delays do not interfere with synchronization.

When a machine comes up and joins the network, it starts a slave time daemon, which will ask the master for the correct time and will reset the machine's clock before any user activity can begin. TEMPO therefore maintains a single network time in spite of the drift of clocks away from each other.

An election algorithm that will elect a new master should the machine running the current master crash, the master terminate (for example, because of a run-time error), or the network be partitioned, ensures that TEMPO provides continuous, and therefore reliable service[4]. However, in the following discussion we will assume that elections do not occur, as we are only concerned with determining the accuracy achieved by the clock synchronization algorithms.

Definitions and General Assumptions

A physical clock generates an approximation, as precise as possible, of t, the universal Galilean time. A real-valued, continuous, and everywhere derivable function $C(t)$ describes its behavior. Let ρ be the absolute value of the maximum drift rate of an actual clock from the universal time; we have:

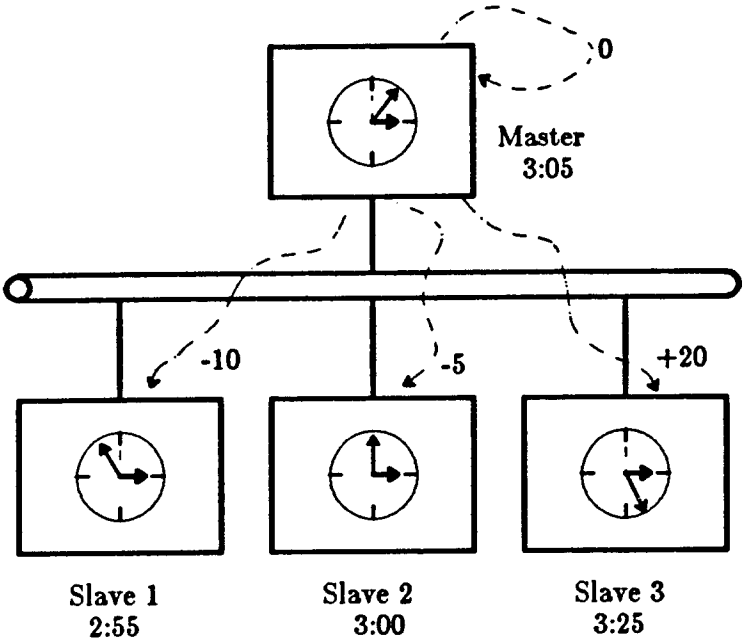
$$1 - \rho \leq \frac{dC(t)}{dt} \leq 1 + \rho \,. \tag{1}$$

Two clocks are said to be *synchronized* at time t_0 if their associated functions have the same value, i.e. if $C_A(t_0) = C_B(t_0)$.

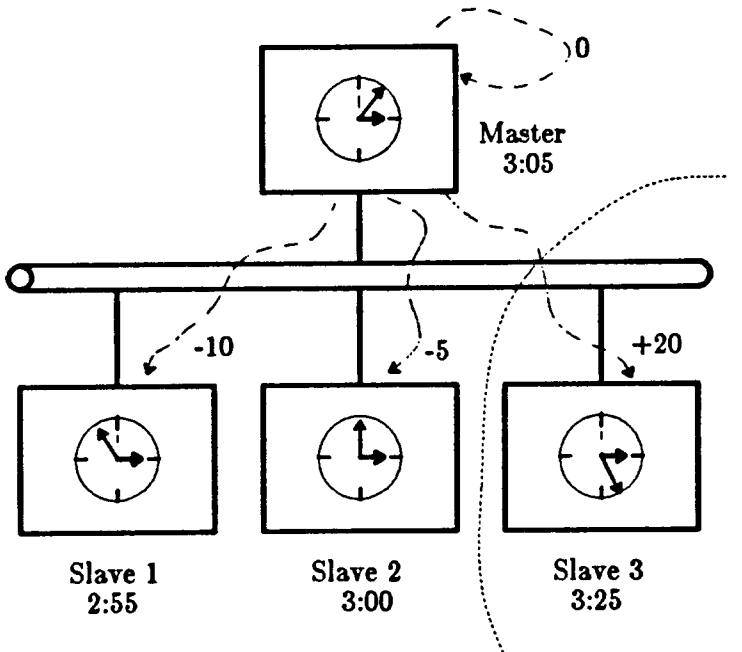
---

‡ TEMPO considers faulty a clock whose value is more than a small specified interval away from those of the majority of the clocks belonging to the machines synchronized by the same master.

The Measurements



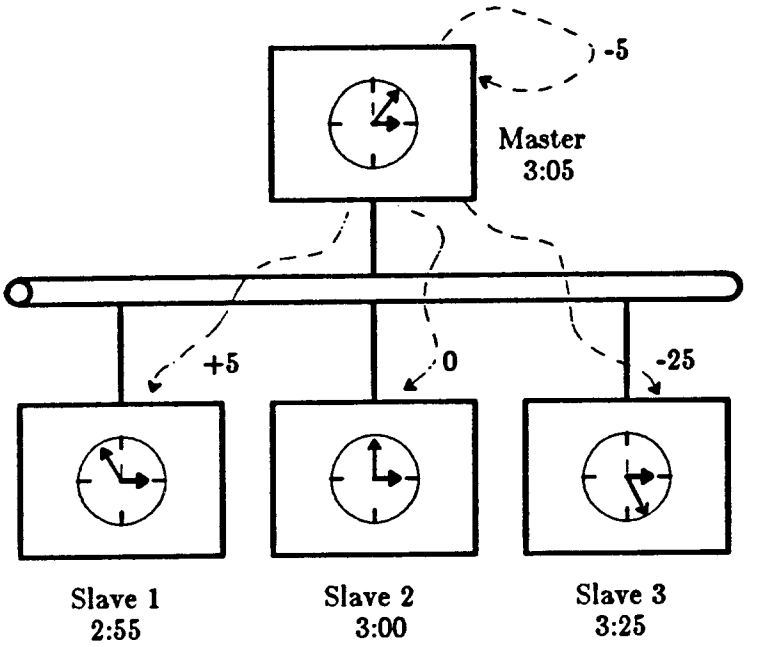
The Computation of the Average



$$Av = \frac{0 - 10 - 5}{3} = -5$$

Figure 1a

The Correction of the Clocks



Clocks are now Synchronized

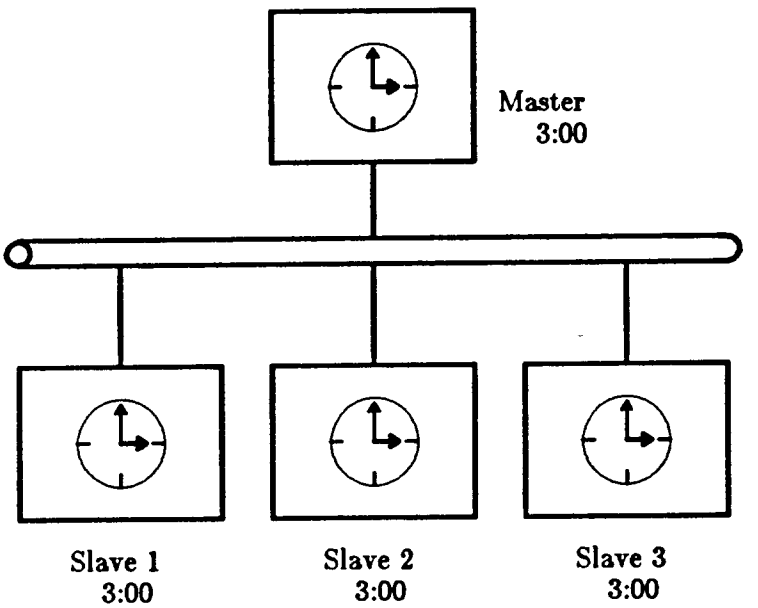


Figure 1b

Let R be a constant. Two or more clocks are *within range R at time* t_0 if the difference between any two of them is bounded by R:

$$\left| C_A(t_0) - C_B(t_0) \right| \leq R \ .$$

Lemma 1:
For $t_1 \geq t_0$:

$$(1-\rho)(t_1-t_0) \leq C(t_1) - C(t_0) \leq (1+\rho)(t_1-t_0) \ .$$

Proof:
Immediate by integrating (1).

Lemma 2:
The absolute value of the relative drift rate of any two clocks satisfying (1), is at most 2ρ:

$$\left| \frac{d(C_A(t) - C_B(t))}{dt} \right| \leq 2\rho \ .$$

Proof:
Let us first assume that clock C_A is fast and clock C_B is slow. From (1) we have:

$$\frac{dC_A(t)}{dt} \leq 1 + \rho \ , \quad \frac{dC_B(t)}{dt} \geq 1 - \rho \ .$$

In this case,

$$\frac{dC_A(t)}{dt} - \frac{dC_B(t)}{dt} \leq 2\rho \ .$$

In the opposite case, in which clock C_A is slow and clock C_B is fast, (1) yields:

$$\frac{dC_A(t)}{dt} - \frac{dC_B(t)}{dt} \geq -2\rho \ .$$

Lemma 2 follows.

A direct consequence of Lemma 2 is that, if two clocks are synchronized at time t_0, at any later time t_1 their values can differ at most by $\pm\ 2\rho(t_1 - t_0)$.

The Clock Difference Measurement Algorithm

Machine A timestamps a message at time $C_A(t_1)$ and sends it to Machine B, which timestamps it at time $C_B(t_2)$ and sends it back†. Upon

† This exchange of messages is implemented in TEMPO using the *TimeStamp* and *TimeStampReply* messages of the DARPA Internet Control Message Protocol

receipt of the message, Machine A reads the time $C_A(t_3)$. Machine A can estimate $\Delta_{AB}(t)$, the difference between its own clock and the clock of Machine B, as

$$\frac{C_A(t_1) + C_A(t_3)}{2} - C_B(t_2) .$$

As indicated, Δ_{AB} is a function of time, but we assume that its variation in the interval $t_3 - t_1$ is so small that we can write:

$$\Delta_{AB}(t_3) = \Delta_{AB}(t_1) = \Delta_{AB} .$$

Also, notice that $\Delta_{AB} = -\Delta_{BA}$.

Theorem 1:
Let $T_{m_{AB}}$ and $T_{m_{BA}}$ be the minimal possible transmission times from A to B and from B to A, respectively‡. Let us fix a bound, $T_M \geq 2\max(T_{m_{AB}}, T_{m_{BA}})$, on the round-trip time, i.e. $C_A(t_3) - C_A(t_1) \leq T_M$. Then, the maximum error in the estimation of Δ_{AB} is:

$$\varepsilon = \frac{T_M - 2\min(T_{m_{AB}}, T_{m_{BA}})}{2} \geq 0 . \qquad (2)$$

Proof:
Let $T_{S_{AB}}$ and $T_{S_{BA}}$ be the actual transmission times from A to B and vice versa. We have:

$$T_{m_{AB}} + T_{m_{BA}} \leq T_{S_{AB}} + T_{S_{BA}} \leq T_M ,$$

and also:

$$\max(T_{S_{AB}}) = T_M - T_{m_{BA}} , \quad \max(T_{S_{BA}}) = T_M - T_{m_{AB}} \qquad (3)$$

for the hypotheses.

We can now compute†:

(ICMP)[10]. As soon as the associated interrupt of the network interface is served, the kernel of a remote machine processes a TimeStamp message by changing its type field to TimeStampReply, writing the clock value in the message, and sending it back without invoking a user process. This implement a variant of an echo protocol. We can therefore consider that the remote time query occurs *instantaneously* at the remote machine at time t_2.

‡ In general $T_{m_{AB}}$ and $T_{m_{BA}}$ will be different, as in the case of a ring network where the information flow travels in the same direction. However, these two times can also be different in a bus network because, for example, of different interrupt structures of the two machines.

† In the actual implementation, several round-trip messages are exchanged and the minimum values of δ_1 and δ_2 are used in the computation of E_{AB}. This reduces the variance of the transmission times in the two directions and provides a better estimate of Δ_{AB}.

$$\delta_1 = C_B(t_2) - C_A(t_1) = -\Delta_{AB} + T_{S_{AB}} ,$$

$$\delta_2 = C_A(t_3) - C_B(t_2) = \Delta_{AB} + T_{S_{BA}} ,$$

and, if E_{AB} is our estimate of Δ_{AB}:

$$E_{AB} = \frac{\delta_2 - \delta_1}{2} = \frac{C_A(t_1) + C_A(t_3)}{2} - C_B(t_2) = \Delta_{AB} + \frac{T_{S_{BA}} - T_{S_{AB}}}{2} \quad (4)$$

From (3) we can derive:

$$-(T_M - 2T_{m_{BA}}) \leq T_{S_{BA}} - T_{S_{AB}} \leq (T_M - 2T_{m_{AB}}) . \quad (5)$$

By substituting (5) into (4), we get:

$$\Delta_{AB} - \frac{T_M - 2T_{m_{BA}}}{2} \leq E_{AB} \leq \Delta_{AB} + \frac{T_M - 2T_{m_{AB}}}{2} . \quad (6)$$

If we define:

$$\varepsilon = \frac{T_M - 2\min(T_{m_{AB}}, T_{m_{BA}})}{2} \geq 0 ,$$

since

$$\varepsilon \geq \frac{T_M - 2T_{m_{AB}}}{2} \quad \text{and} \quad \varepsilon \geq \frac{T_M - 2T_{m_{BA}}}{2}$$

for the definition of T_M, then the theorem follows:

$$\left| E_{AB} - \Delta_{AB} \right| \leq \varepsilon . \quad (7)$$

If the estimate E_{AB} is used to synchronize the clock of Machine B, the two machines' clocks are, upon synchronization, within range ε.

Corollary 1:
The lower bound for the error ε is:

$$\varepsilon \geq \left| T_{m_{AB}} - T_{m_{BA}} \right| .$$

Proof:
Immediate by substituting into (2) the expression for T_M.

Corollary 2:
The measurement algorithm allows a machine to compute the clock difference between any two other machines with maximum error 2ε.

Proof:
Let us suppose that machine A sends clock difference measurement messages to any two machines, for instance machines B and C, then:

$$\Delta_{AB} = C_A(t) - C_B(t) , \quad E_{AB} = \Delta_{AB} \pm \varepsilon ,$$

$$\Delta_{AC} = C_A(t) - C_C(t) , \quad E_{AC} = \Delta_{AC} \pm \varepsilon ,$$
$$\Delta_{BC} = \Delta_{AC} - \Delta_{AB} , \quad E_{BC} = E_{AC} - E_{AB} .$$

It follows:

$$E_{BC} = \Delta_{AC} - \Delta_{AB} \pm 2\varepsilon = \Delta_{BC} \pm 2\varepsilon .$$

The Synchronization Algorithm

The master, using the clock difference measurement algorithm, computes the time differences between its clock and the clocks of slave machines. A fault-tolerant averaging function is then applied to these differences. It selects the largest sets of clocks that do not differ from each other more than a small quantity γ and averages the differences of these clocks. For instance, in the example of Figures 1a and 1b, assuming that γ is 10 minutes, the fault-tolerant function selects the set consisting of the clock of the Master, the clock of Slave 1, and that of Slave 2. This averaging function prevents malfunctioning clocks as well as clocks with abnormally large drift rates from adversely affecting other clocks. Notice, however, that the synchronization algorithm produces the appropriate correction value for every clock. Clocks that are not selected by the fault-tolerant function are considered faulty. Last, the master asks each slave to correct its clock by a quantity equal to the difference between the average value and the previously measured difference between the clock of the master and that of the slave. This process is repeated every T seconds.

For TEMPO to be reliable, it is necessary that all properly functioning clocks be within γ seconds when the master starts a synchronization round. The constant γ is therefore chosen as a function of the clock drift rate; the interval between synchronization rounds, T; and the measurement errors as derived in Theorem 3 below.

Theorem 2:
If the master, using the synchronization algorithm described above synchronizes a number of machines, then any two non-faulty clocks are, once the synchronization is performed, within range 4ε.

Proof:
Let Q be the set of machines selected by the fault-tolerant averaging function. The average of the measurements is then:

$$\frac{1}{|Q|}\sum_{J\varepsilon Q} E_{AJ} = \frac{1}{|Q|}\sum_{J\varepsilon Q} \Delta_{AJ} \pm \left[\frac{|Q| - 1}{|Q|}\right]\varepsilon , \qquad (8)$$

where we have assumed that the clock of the master A is also non-faulty‡ and $\Delta_{AA} = 0$ with no error by definition.

‡ This is not a necessary assumption. The algorithm and the derivations will con-

If we use the symbol $\bar{\Delta}$ for $\frac{1}{|Q|}\sum_{J\varepsilon Q}\Delta_{AJ}$ in order to simplify the notation, we can rewrite (8) as:

$$\left| \frac{1}{|Q|}\sum_{J\varepsilon Q} E_{AJ} - \bar{\Delta} \right| \leq \varepsilon' \qquad (9)$$

with $\varepsilon' = \varepsilon$.

The correction performed on the clock of machine K is:

$$c_K = \frac{1}{|Q|}\sum_{J\varepsilon Q} E_{AJ} - E_{AK} ,$$

from which, by adding the quantity $\Delta_{AK} - \bar{\Delta}$, and for (7) and (9) we obtain:

$$\left| c_K + \Delta_{AK} - \bar{\Delta} \right| \leq \left| \frac{1}{|Q|}\sum_{J\varepsilon Q} E_{AJ} - \bar{\Delta} \right| + \left| \Delta_{AK} - E_{AK} \right| \leq \varepsilon' + \varepsilon .$$

Let us represent with Δ'_{BC} the difference between the clocks of machines B and C after the correction is made:

$$\Delta'_{BC} = (\Delta_{AC} + c_C) - (\Delta_{AB} + c_B) .$$

By adding and subtracting $\bar{\Delta}$ we can write:

$$\Delta'_{BC} = (\Delta_{AC} + c_C - \bar{\Delta}) - (\Delta_{AB} + c_B - \bar{\Delta}) ,$$

and also:

$$\left| \Delta'_{BC} \right| \leq \left| c_C + \Delta_{AC} - \bar{\Delta} \right| + \left| \bar{\Delta} - c_B - \Delta_{AB} \right| \leq 2\varepsilon' + 2\varepsilon = 4\varepsilon$$

which completes the proof.

The following theorem summarizes the previous results:

Theorem 3:

If a machine measures the Δ's for a set of other machines and synchronizes them every T seconds, then, at any time, all non-faulty clocks are within range $4\varepsilon + 2\rho T$.

Proof:

The first item, 4ε, as per Theorem 2, accounts for the inaccuracy of synchronization after the clocks have been reset. The second, as per Lemma 2, accounts for the maximum drift of any two clocks during the time between two subsequent synchronizations.

---

tinue to be valid whether or not the master's clock is selected by the fault-tolerant averaging function. Refer, however, to the next section of this paper for a brief discussion of the types of faults that TEMPO can tolerate.

Discussion

It is important to notice that in the derivation of the bounds on the time accuracy we have made no assumption whatsoever about the statistical distribution of the transmission times between two machines, nor have we assumed that these distributions are the same in the two communication directions.

It should also be noted that the requirements on the maximum round-trip time T_M can be verified by the master, in the notation used above, by computing $C_A(t_3) - C_A(t_1)$. Even though messages can be arbitrarily delayed, the master is always able to reject measurements that do not satisfy the conditions of Theorem 1.

In our implementation of TEMPO for the Ethernet local area network, we have chosen a value of 20 milliseconds for T_M. Although the Digital Equipment VAX Hardware Handbook states that ρ can be as high as 10^{-4}, we have verified, using a high-resolution frequency meter, that the clocks of the VAX's used in our experiments display drift rates smaller than 2 parts in 10^5. Since the minimum transmission delay from machine to machine can be estimated to be 5 milliseconds (including kernel protocol handling and the scheduling delays of the master process), and since TEMPO synchronizes the clocks every 4 minutes, the maximum error in Theorem 3 is 30 milliseconds.

Let us call ε_{AB} the *actual* error in the measurement of the clock difference between machines A and B. From (6) we have: $-\varepsilon \leq \varepsilon_{AB} \leq +\varepsilon$. Therefore, the actual quantity that corresponds to ε' in (9) is, for (8), $\frac{1}{|Q|}\sum_{J\varepsilon Q}\varepsilon_{AJ}$ that is the average of the actual errors of the measurements between the master A and the other machines in the set Q. As such, by the Strong Law of Large Numbers, this quantity converges in probability to the mean of the random variable that models the measurement errors. Under the condition of identically distributed transmission times in the two communication directions, which is satisfied in the case of the Ethernet[†], this mean, as can be recognized in (6), is zero. While according to Theorem 3 the first component of the global error can be as large as 4ε, the algebraic manipulations in the proof of Theorem 2 show that it can be separated into two parts, one of which, $2\varepsilon'$, for what we have just seen, should be very small.

In measurements taken in our environment, where the time daemons synchronized the clocks of about 15 machines, we rarely found the time

[†] See also footnote to Theorem 1.

difference between clocks to be larger than 25 milliseconds, with the mean between 18 and 20 milliseconds. Since the drift rate of the clocks makes them diverge at most 10 milliseconds in 4 minutes, we estimated that the synchronization inaccuracies due to the error described in Theorem 2 amount to about 10 milliseconds on the average.

As previously observed, a clock is considered faulty if it is not selected by the fault-tolerant averaging function. Therefore, great attention must be paid to the appropriate choice for the value of γ. If γ is too small, only a few clocks may be selected; if it is too large, malfunctioning clocks can reduce the precision of the synchronized time. In both cases, the reliability of TEMPO decreases. Since our measurements showed that most clocks do not diverge more than 20 milliseconds from each other, we set γ equal to 20 milliseconds.

The fault-tolerant averaging function may reject a clock measurement for any of three reasons. First, there may be a hardware malfunction. Second, a clock difference measurement may follow a clock adjustment with an above-average error. Finally, in an improperly set-up machine, a series of high-priority interrupts may prevent the operating system from servicing lower-priority timer clock interrupts, causing that machine's clock to slow down. Given that TEMPO was designed for an environment where Byzantine faults are highly improbable, the synchronization algorithm can tolerate $\frac{N-1}{2}$ faults. However, it should be noted that the clock of the master, which is not considered more important than any other clock by the fault-tolerant averaging function, may cause the clock difference measurement algorithm to fail if it is double-faced.

Comparison with Previous Work

Although Tempo is a distributed program, it uses a centralized approach in directing the synchronization activities. Fault-tolerance is achieved by not giving a privileged role to the master's clock in the synchronization algorithm and by providing an election algorithm that elects a new master should the old one terminate. Our approach therefore contrasts with other existing algorithms that adopt a fully distributed approach to fault-tolerance.

It is difficult to compare the various clock synchronization algorithms because, as observed by Lamport and Melliar-Smith[7], different algorithms require different methods of reading clocks and each method generates a different error. In addition, the various authors describe the bounds on their algorithms using parameters not always easily convertible to those of

our system of variables. However, in general, the errors in clock synchronization, as in Theorem 3, depend on the *uncertainty* in the elapsed time between the generation and the receipt of a message and on the time between synchronization rounds.

In the remainder of this section, in order to compare the bounds on the accuracy of different algorithms, we make the following three additional assumptions: 1) there are $N=3F+1$ machines, where F is the number of machines with faulty clocks; 2) the transmission time between any two machines is equally distributed; and 3) the message delivery time is in the range $[\tau-\eta, \tau+\eta]$, where τ is the median delay time and η is the uncertainty. Also, notice that our purpose is to point out the main advantages of our algorithm over some alternative clock synchronization methods rather than to comprehensively review the literature in this area.

Lundelius and Lynch[9] describe an algorithm that executes in a series of rounds; each round is started when a clock reaches a certain predefined value. When this happens, a machine broadcasts that value to all other machines. Meanwhile, it collects within a particular bounded amount of time measured on its own clock, messages from other machines. Then, each machine computes the correction for its clock using a fault-tolerant averaging function. The bound analysis shows that clocks can be synchronized as closely as $4\eta+4\rho T$, but the authors suggest that, with a slight modification of their algorithm, they can reduce the second term to $2\rho T$.

The algorithm designed by Halpern et al.[5] is also based on the periodic broadcasting of clock values. In their method however, a machine that receives a message with a value that its clock has not reached yet, updates the clock to that value and broadcast the corresponding message. This algorithm generates an error of $\tau+\eta+2\rho T$.

The three algorithms introduced by Lamport and Melliar-Smith[6], CON, COM, and CSM, are based on broadcast as well and achieve the following accuracy respectively: $2N\eta+N\rho T$, $2(N+1)\eta+\rho T$, and $\frac{N+17}{3}\eta+\rho T$.

Although Lamport and Melliar-Smith do not give the synchronization error in a form comparable to ours –they analyze how closely in real time clocks reach the same value whereas we measure how close clocks are at the same real time–, the two quantities appear to be similar.

While it is true that most communication protocols are designed to provide an upper bound on the communication time, perhaps by abnormally terminating the transmission after a number of retries, it is also true that the resulting variance in the transmission times can be much larger than the average transmission time. A unique feature of our algorithm is that it

can bound the round-trip time, despite the high variance in transmission times, by rejecting those measurements that do not satisfy the requirements of Theorem 1. In fact, under the assumptions introduced above, if we call T_m the minimum transmission time, we have:

$$\tau - \eta = T_m , \quad \tau + \eta = T_M - T_m ;$$

and

$$\tau = \frac{T_M}{2} , \quad \eta = \frac{T_M - 2T_m}{2} .$$

By comparing the expression for η with (2), we can rewrite the result of Theorem 3 as:

$$4\eta + 2\rho T .$$

Although the formula for the accuracy of our algorithm is the same as the one for the algorithm of Lundelius, our η is much lower than theirs. Using for the parameters the values we have introduced earlier in this section, we obtain $\tau = 10$ milliseconds and $\eta = 5$ milliseconds. In the case of other algorithms, η is proportional to the standard deviation of the transmission times, which for the Ethernet can be rather large when messages collide. When clocks are synchronized —or almost synchronized— the simultaneous broadcasting of messages that occurs in the algorithms, may cause numerous collisions, increasing both the median transmission time τ and the uncertainty η. Therefore in an Ethernet environment, we would expect that our algorithm achieve significantly better synchronization accuracy. In a non-Ethernet environment, for instance a ring or point-to-point network, we would still expect that η of the other algorithms would be larger than our η, though the difference between the two may be smaller.

Algorithms COM and CSM were developed in the framework of Byzantine clock synchronization and both require about N^{F+1} messages. Algorithm CON and the algorithms of Lundelius and Halpern require in the worst case about N^2 messages. TEMPO, in contrast with the other algorithms, employs for each synchronization round only a linear number of messages. However, unlike TEMPO which needs an election mechanism to ensure that a new master be elected in case the current one crashes or the network partitions, those algorithms are inherently fault-tolerant. Our choice is motivated by the fact that in our computing environment the kind of faults that require the intervention of the election procedure are rare. We have followed a design principle[8] that calls for simplicity in the most common situations and confines complexity and high costs with unusual conditions.

Conclusions

We have discussed the upper and lower bounds on the accuracy achieved by the clock synchronization algorithms of TEMPO which is distributed with Berkeley UNIX 4.3BSD. TEMPO keeps the clocks of VAX computers in a local area network synchronized with an accuracy comparable to the resolution of single machine clocks. Comparison with other clock synchronization algorithms shows that TEMPO, in an environment with no Byzantine faults, may achieve better synchronization at a lower cost.

Acknowledgments

The authors would like to thank Domenico Ferrari and Mike Karels for their valuable advice during the development and implementation of these algorithms.

References

[1]. "Adjtime System Call," *UNIX Programmer's Manual (Section 2)*, 4th Berkeley UNIX Distribution Release 3, February 1985.

[2]. R. Gusella and S. Zatti, "TEMPO - A Network Time Controller for a Distributed Berkeley UNIX System," *Distributed Processing Tech. Comm. Newsletter*, vol. 6 NoSI-2, pp. 7-15, IEEE, June 1984.

[3]. R. Gusella and S. Zatti, "The Berkeley UNIX 4.3BSD Time Synchronization Protocol: Protocol Specification," *Report No. UCB/CSD 85/250*, University of California, Berkeley, June 1985.

[4]. R. Gusella and S. Zatti, "An Election Algorithm for a Distributed Clock Synchronization Program," *IEEE 6th International Conference on Distributed Computing Systems*, pp. 364-371, Boston, May 1986.

[5]. J. Halpern et al., "Fault-Tolerant Clock Synchronization," *Proceedings of the 3th ACM Annual Symposium on Principles of Distributed Computing*, pp. 89-102, Vancouver, August 1984.

[6]. L. Lamport and P.M. Melliar-Smith, "Byzantine Clock Synchronization," *Proceedings of the 3th ACM Annual Symposium on Principles of Distributed Computing*, pp. 68-74, Vancouver, August 1984.

[7]. L. Lamport and P.M. Melliar-Smith, "Synchronizing Clock in the Presence of Faults," *Journal of the ACM*, vol. 32, pp. 52-78, January 1985.

[8]. B. Lampson, "Hints for Computer System Design," *Proceedings of the 9th SOSP, Operating System Review*, vol. 17, pp. 33-48, ACM, October 1983.

[9]. J. Lundelius and N. Lynch, "A New Fault-Tolerant Algorithm for Clock Synchronization," *Proceedings of the 3th ACM Annual Symposium on Principles of Distributed Computing*, pp. 75-88, Vancouver, August 1984.

[10].J. Postel (ed.), "Internet Control Message Protocol - DARPA Internet Program Protocol Specification," *RFC 792*, USC/Information Science Institute, September 1981.